Certificate of Correction Change: Column 3, line 36, after the word "According" insert therefor --to--;

Certificate of Correction Change: Column 3, line 48, delete "back-end" and insert therefor --backend--.

Column 3, lines 36-52 are reproduced as follows:

According to a third embodiment of the present invention, a display system is provided which includes first and second parallel backend pipelines. A multi-format frame buffer memory is included having on-screen and off-screen memories each operable to simultaneously store data in graphics and video formats. A dual aperture port is provided for receiving both graphics and video data as directed by an address associated with each word of data received. Circuitry for writing is included for writing a word of video or graphics data into a selected one of the on-screen and off-screen areas of the multi-frame buffer. Memory control circuitry controls the transfer of data between the first and second backend pipelines and the frame buffer. The system further includes a display unit and overlay control circuitry for selecting for output to the display unit between data provided by the first backend pipeline and data provided by the second backend pipeline.

Certificate of Correction Changes: Column 5, line 45, delete both occurrences of "back-end" and insert therefor --backend--.

Column 5, lines 39-51 are reproduced as follows:

FIG. 2 is a more detailed functional block diagram of VGA controller 105. The primary circuitry blocks of VGA controller 105 include video front-end video pipeline 200, memory (frame buffer) control circuitry 201, CRT/window control circuitry 202, video window control registers 203, video backend pipeline 204 and graphics backend pipeline 205. VGA controller 105 further includes a CPU interface 206 for exchanging instructions and data via a PCI or VL bus, such as local bus 103 in system 100, with CPU 101. A write buffer 207 and conventional graphics controller 208 allow CPU 101 to directly control data within frame buffer 107 via memory control circuitry 201.

Certificate of Correction Change: Column 5, line 66, after "211" insert therefor --is--;

Certificate of Correction Change: Column 6, line 6, delete "back-end" and insert therefor --backend--.

Column 5, lines 66 through column 6, line 13 are reproduced as follows:

Data which is input through the video port 211 is address-free. In this case, video window controls 213 generates the required addresses to either the on-screen memory area or the off-screen memory as a function of display location for the video window. In the preferred embodiment, window controls 213 generate addresses using the same video control registers 203 used to control retrieval of the video in the backend pipeline (i.e., the screen x and y position registers 500 and 501 discussed below in conjunction with FIG. 5). When data is being received through both the CPU interface 206 and the VPORT 211 simultaneously, the data is interleaved into memory with the two write buffers 207 and 217 buffering the data such that neither stream is interrupted or forced into a wait state at the source component (i.e., bus 103 or video source 104).

Certificate of Correction Change: Column 6, line 17, after "generated." insert therefor --)-- ;

Certificate of Correction Change: Column 6, line 25, after "will" insert therefor --be--;

Certificate of Correction Change: Column 6, line 25, after "discussed" delete "further".

Column 6, lines 14-29 have been reproduced as follows:

It should be noted at this point that frame buffer 107 includes at least two different data areas or spaces to which data can be directed by the given address (either CPU 103 or controls 213 generated.) Each space can simultaneously store graphics or video data depending on the selected display configuration. The on-screen area corresponds to the display screen; each pixel rastered out of a given pixel location in the on-screen area defines a corresponding screen pixel. The off-screen area is used to store data defining a window for selectively overlaying the data from the on-screen memory, fonts and other data necessary by controller 105. Further, as will be discussed below,

both graphics and video data may be rastered from frame buffer 107 and passed through video backend pipeline 204 while only graphics data is ever passed through graphics backend pipeline 205.

Certificate of Correction Change: Column 6, line 45, after "through" delete "the video playback data is passed through".

Column 6, lines 33-48 have been reproduced as follows:

For example, CPU 103 may write a static graphics background into part of the on-screen memory with the remaining "window" in the on-screen memory area filled with playback video data. "Playback" video data can be either (1) live video data input from the VPORT; (2) YUV (video) data written through interface 206 by CPU 103; or (3) true color (5:5:5, 5:6:5, or 8:8:8) RGB graphics data (for example animation graphics data) written in through either the VPORT or interface 206. Similarly, a playback video background and a window of graphics data may be written into the on-screen area. In each of these cases, the data is rastered out as the display is without overlay; the video playback data is passed through the video backend pipeline 204 as a function of display position by controls 202 and the graphics data passed through the graphics backend pipeline 250.

Certificate of Correction Change: Column 6, line 66, delete "of" and insert therefor --to--;

Certificate of Correction Change: Column 6, line 67, delete ", the" and insert therefor --. The--.

Column 6, lines 49-67 have been reproduced as follows:

Windows of data retrieved from the off-screen memory can be retrieved and used to occlude a portion of the data being rastered out of the on-screen memory. For example, a window of playback data can be stored in the off-screen memory and a frame of static graphics data (either true color data or indices to CLUT 234) stored in the on-screen memory. In this case, the static graphics are rastered out of the on-screen memory without interruption and passed through the graphics backend pipeline 205. The window of data in the off-screen memory is rastered out only when the display position for the

window has been reached .by the display raster and is passed through video backend pipeline 204. As discussed below, data from the video backend pipeline 204 can then be used to selectively occlude (overlay) the data being output from the graphics backend pipeline 205. A window of static graphics data (true color or indices to the CLUT 234) can be stored in off-screen memory and used to overlay playback video from the on-screen memory. The playback video data is passed through the video backend pipeline 204 and the window of static graphics data is passed through the graphics backend pipeline 205.

Certificate of Correction Change: Column 7, line 10, delete "back-end" and insert therefor --backend--;

Certificate of Correction Change: Column 7, line 11, delete "back-end" and insert therefor --backend--;

Certificate of Correction Change: Column 7, line 11, delete "pipe-line" and insert therefor --pipeline--.

Column 7, lines 4-11 have been reproduced as follows:

Bit block transfer (BitBLT) circuitry 209 is provided to allow blocks of graphics data within frame buffer 107 to be transferred, such as when a window of graphics data is moved on the display screen by a mouse. Digital-to-analog converter (DAC) circuitry 210 provides the requisite analog signals for driving display 106 in response to the receipt of either video data from video backend pipeline 204 or graphics data from backend pipeline 208.

Certificate of Correction Change: Column 7, line 40, after "therefrom" delete ")";

Certificate of Correction Change: Column 7, line 42, delete "back-end" and insert therefor --backend--.

Column 7, lines 12-44 have been reproduced as follows:

In implementing the operations discussed above, video front-end pipeline 200 can receive data from two mutually exclusive input paths. First, in the "playback mode," playback (non-real time) data may be received via the PCI bus through CPU interface 206. Second, in the "overlay emulation mode" either real-time or playback video may be

received through the video port interface 211 (in system 100 video port interface 211 is coupled to bus 109 when real-time data is being received). The selection of video from the PCI bus or video from video port interface 211 is controlled by a multiplexer 212 under the control of bits stored in a video front-end pipeline control register within video control registers 203. In the playback mode, either CPU 101 or a PCI bus master controlling the PCI bus provides the frame buffer addresses allowing video front-end pipeline 200 to map data into the frame buffer separate and apart from the graphics data. In the overlay emulation mode, overlay input window controls 213 receives framing signals such as VSYNC and HSYNC, tracks these sync signals with counters to determine the start of each new frame and each new line, generates the required addresses for the real-time video to the frame buffer space using video window position data received from window controls 222 (as discussed above, in the preferred embodiment, video data is always retrieved from either the on-screen on off-screen memory and passed through video back-end pipeline 204 as a function of display position) and thus the position data from controls 222 is used to both write data to memory and retrieve data therefrom. In general, overlay input video control windows are controlled by the same registers which control the backend video pipeline 204, although the requisite counters and comparators are located internal to overlay input video control circuitry 213.

Video front-end pipeline 200 also includes encoding circuitry 214 that is operable to truncate 16-bit YUV 422 data into an 8-bit format and then pack four such 8-bit encoded words into a single 32-bit word which is then written into the video frame buffer space of frame buffer 105. Conversion circuitry 215 is operable to convert RGB 555 data received from either the CPU interface 206 and the PCI bus or VPORT I/F 211 into YCrCb (YUV) data prior to encoding by encoding circuitry 214. Conversion circuitry 215 allows graphics data (for example in a 5:5:5 or 5:6:5 format) to be introduced through the VPORT or graphics data to be converted, packed and stored in a YUV format

in the off-screen memory space by CPU 101. For a more complete description of encoder 214 and the associated decoder 225 of video pipeline 204, reference is now made to incorporated copending coassigned application Ser. No. 08/223,845. The selection and control of the encoding circuitry 214 and conversion circuitry 215 is implemented through multiplexing circuitries 212 and 216, each of which are controlled by bits in the video control registers. Finally, video front-end pipeline 200 includes a write buffer/FIFO 217 which in one embodiment acts as a write buffer and in an alternate embodiment acts as a FIFO for the video backend pipeline 204. In embodiments where buffer 217 acts as a write buffer for then Y, zooming on the backend, as discussed below is by replication. In embodiments where buffer 217 operates as a FIFO, then the VPORT and front and end color conversion by converter circuitry 215 are not used for writing data to frame buffer 107.

Certificate of Correction Change: Column 8, line 33, delete "back-end" and insert therefor --backend--;

Certificate of Correction Change: Column 8, line 49, delete "back-end" and insert therefor --backend--.

Column 8, lines 33-61 have been reproduced as follows:

Video backend pipeline 204 receives a window of graphics video data defining a display window from the on-screen or off-screen spaces in frame buffer 107 through a pair of first-in/first-out memories 223 and 217 (in embodiments where buffer 217 is acting as FIFO B). In the preferred embodiment, each FIFO receives the data for every other display line of data being generated for display on the display screen. For example, for a pair of adjacent lines n-1 and n+1 in memory (although not necessarily adjacent on the display) for the display window, FIFO 223 receives the data defining window display line n-1 while FIFO 224 receives the data defining window display line n+1. When buffer 217 is used as FIFO B, writes through video front end pipeline 200 are made through write buffer I 207 and multiplexer 235. Alternatively, if buffer 217 is used as write buffer II, then FIFO B is not implemented and only a single stream is processed by video backend pipeline 204 (no Y interpolation is performed and Y expansion is by replication). As will be discussed further below (assuming both FIFO A and FIFO B are

being used), one or more display lines, which falls between line n-1 and line n+1, may be selectively generated by interpolation. Decoder circuitry 225 receives two 32-bit packed words (as encoded by encoder 214), one from each adjacent scan line in memory, from FIFOs 223 and 217. Each 32-bit word, which represents four YCrCb pixels, is expanded and error diffused by decoder 225 into four 16-bit YCrCb pixels. In modes where video data is stored in the frame buffer in standard 555 RGB or 16 YCrCb data formats, decoder block 225 is bypassed.

Certificate of Correction Change: Column 8, line 62, delete "Back-end" and insert therefor --Backend--.

Column 8, beginning line 62 through column 9, line 7 has been reproduced as follows:

Backend video pipeline 204 further includes a Y interpolator 226 and X interpolator 227. In the preferred embodiment, during Y zooming (expansion) Y interpolator 226 accepts two vertically adjacent 16-bit RGB or YCrCb pixels from the decoder 225 and calculates one or more resampled output pixels using a four subpixel granularity. X interpolator 227 during X zooming (expansion) accepts horizontally adjacent pixels from the Y interpolator 226 and calculates one or more resampled output pixels using a four subpixel granularity. For data expansion using line replication, Y interpolator 226 is bypassed. Y interpolator 226 and X interpolator 227 allow for the resizing of a video display window being generated from one to four times.

Certificate of Correction Change: Column 9, line 11, after "through" insert --the--;

Certificate of Correction Change: Column 9, line 11, after "pipeline" insert --,--.

Column 9, lines 8-12 have been reproduced as follows:

The output of X interpolator 227 is passed to a color converter 228 which converts the YCrCb data into RGB data for delivery to output multiplexer 304. To reiterate, if graphics data is passed through the video pipeline, converter 228 is not used.

Certificate of Correction Change: Column 9, line 13, delete "Back-end" and insert therefor --Backend--;

Certificate of Correction Change: Column 9, line 24, delete "back-end" and insert therefor --backend--.

Column 9, lines 13-26 have been reproduced as follows:

Backend video circuitry 204 further includes pipeline control circuitry 229, overlay control circuitry 230 and output multiplexer 231. Pipeline control circuitry 239 controls the reading of data from video FIFOs 223 and 217, controls the generation of interpolation coefficients for use by X and Y interpolators 226 and 227 to resize the video window being pipelined, and times the transfer of data through the pipeline. Overlay control circuitry 230 along with control circuitry 202, controls the output of data through output multiplexer 231, including the overlay of the video window over the graphics data output through the graphics backend pipeline 205. A pixel doubler is provided to double the number of pixels being generated such that a 1280x1024 display can be driven.

Certificate of Correction Change: Column 9, line 27, delete "back-end" and insert therefor --backend--;

Certificate of Correction Change: Column 9, line 34, after "directly" insert --to--.

Column 9, lines 27-41 have been reproduced as follows:

Graphics backend pipeline 205 includes a first-in/first-out memory 232, attribute controller 233, and color look-up table 234. Each 32-bit word output from graphics FIFO 232 is serialized into either 8-bit, 16-bit or 24-bit words. The 8-bit words, typically composed of an ASCII code and an attribute code, are sent to attribute controller 233. When 16-bit and 24-bit words, which are typically color data, are serialized, those words are sent directly to overlay controls 230. Attribute controller 233 performs such tasks as blinking and underlining operations in text modes. The eight bits output from attribute controller 233 are pseudo-color pixels used to index CLUT 234. CLUT 234 preferably outputs 24-bit words of pixel data to output multiplexer 231 with each index. When video data is being pipelined through graphics backend pipeline 205 from the on-screen memory, CLUT 234 is bypassed.

Certificate of Correction Change: Column 9, line 42, delete "are";

Certificate of Correction Change: Column 9, line 45, delete "back-end" and insert therefor --backend--;

Certificate of Correction Change: Column 9, line 58, delete "back-end" and insert therefor --backend--.

Column 9, lines 42-67 have been reproduced as follows:

The eight bit pseudo-color pixels output from attribute controller 233 are also sent to overlay controls 230. In the preferred embodiment, data is continuously pipelined from on-screen memory through graphics backend pipeline 205 to the inputs of output multiplexer 231. Window data from off-screen memory however is only retrieved from memory and pipelined through video backend pipeline 204 when a window is being displayed. In other words, when a window has been reached, as determined by control bits set by CPU 101 in VW control registers 222, video window display controls 222 generate addresses to retrieve the corresponding data from the off-screen memory space of frame buffer 107. Preferably, video FIFOs 223 and 224 are filled before the raster scan actually reaches the display window such that the initial pixel data is available immediately once the window has been reached. In order to insure that graphics memory data continues to be provided to graphics backend pipeline 205, video window display controls 222 "steal" page cycles between page accesses to the graphics memory. It should be noted that once the window has been reached the frequency of cycles used to retrieve window data increases over the number used to fill the video FIFOs when outside a window. When the frequency of window page accesses increases, video window display controls 222/arbiter 221 preferably "steal" cycles from page cycles being used to write data into the frame buffer.

Certificate of Correction Change: Column 10, line 5, delete "back-end" and insert therefor --backend--;

Certificate of Correction Change: Column 10, line 7, delete "back-end" and insert therefor --backend--.

Column 10, lines 1-14 have been reproduced as follows:

FIG. 3 is a more detailed functional block diagram emphasizing the circuitry controlling the overlay of data from graphics pipeline 205 with window data from video pipeline 204. As discussed briefly above, the inputs to output multiplexer 231 are data from video backend pipeline 204 (pixel doubler 237), 16 or 24-bit color data directly from graphics backend pipeline 205 serializer 236 and 24-bit color data from the color look-up table 234. The output of data to DAC 210 through output multiplexer 231 is controlled by a latch 301 clocked by the video clock (VCLK). The remaining circuitry shown in FIG. 3, which will be discussed in further detail below, provide the necessary control signals to the control inputs of output multiplexer 231 to select between the video and graphics pipelines.

Certificate of Correction Change: Column 10, line 19, delete "circuit" and insert therefor --circuitry--;

Certificate of Correction Change: Column 10, line 29, delete "24-bits of" and insert therefor --24--.

Certificate of Correction Change: Column 10, line 30, delete "blue/index" and insert therefor --blue index--.

Column 10, lines 15-32 have been reproduced as follows:

The graphics pseudo-pixels output from attribute controller 233 and the 16-bit or 24-bit graphics or video data output directly from serializer 236 are provided to the inputs of color comparison circuitry 302. Also input to color comparison circuitry 302 are 16 or 24-bit overlay color key bits stored in overlay color key register 303. Overlay color key register 303 resides within the address space of, and is loaded by, CPU 101. Depending on the mode, color comparison circuitry 302 compares selected bits from the overlay color key register 303 with either the 8 bits indexing look-up table 234 in the color look-up table mode (pseudo-color mode) or the 16-bits (24-bits in the alternate embodiment) passed directly from serializer 236. It should be noted that in the illustrated embodiment, overlay color key register 303 holds 24 overlay color key bits, eight each for red, green, and blue index comparisons. The specific overlay color key bits compared with the input graphics data are provided in Table I:

Certificate of Correction Change: Column 10, line 51, delete "in," and insert therefor --in--;

Certificate of Correction Changes: Column 10, line 60, delete both occurrences of "back-end" and insert therefor --backend--.

Column 10, lines 42-62 have been reproduced as follows:

As shown in FIG. 4A, a first embodiment of color comparison circuitry 303 performs the comparisons set forth in Table I as a set of XNOR operations in series with an AND operation. FIG. 4A depicts first comparison circuitry 400 for comparing the 8-bits of graphics pixels received in the look-up table mode from attribute controller 233 with the 8-bit blue/index overlay key bits being held in overlay key register 303. Second comparison circuitry 401, performs the required comparisons of Table I for the 16-bit data or 24-bit received from serializer 236, in either a 5:5:5, 5:6:5, or 8:8:8 format. An overlay register 402 includes a bit loaded by CPU 101 which is used by a selector 403, depending on the mode, to select for output, either the result of the comparisons being made by comparison circuitry 400 in the color look-up table mode or the results of the comparisons being made by comparison circuitry 401. In the illustrated embodiment, color comparison circuitry 303 processes data on a pixel-by-pixel basis and is resynchronized with both the graphics backend pipeline 205 and the video backend pipeline 204 by having its outputs latched to the video clock (VCLK) by latches 404.

Certificate of Correction Change: Column 11, line 14, after "which" delete ",";

Certificate of Correction Change: Column 11, line 27 [sic -28] delete "are";

Certificate of Correction Change: Column 11, line 28 [sic -29], after "multiplexer" insert --405--.

Column 11, lines 6-46 have been reproduced as follows:

Pixel position comparison circuitry 305 includes three inputs coupled respectively to video window 1 position control circuitry 308, CRT position control circuitry 309 and video window 2 position control circuitry 310. In the illustrated embodiment, CRT position controller 309 is located within CRT controller 220 while video window 1 position control circuitry and video window 2 position control circuitry 310 are located within video display window controls 222 (FIG. 2). CRT position control circuitry 309

includes counters which track the position of the current pixel being generated for display. In the preferred embodiment, CRT position control circuitry 309 includes at least an x-position counter which tracks the generation of each pixel along a given display line and a y-position counter which tracks the generation of each display line in a screen. The x-position counter may for example count pixels by counting each VCLK period between horizontal synchronization signal (HSYNC) controlling display unit 106. The y-position counter may for example count each HSYNC signal occurring between each vertical synchronization signal (VSYNC) controlling the screen generation on display unit 106. FIG. 4B is an alternate embodiment of the color comparison circuitry of FIG. 3. In a first mode, 8 bits from attribute controller 233 are passed through multiplexer 405 to comparator 406. Comparator 406 compares the received eight bits with an 8-bit color key in color key register 408; when the received 8-bits equal the 8-bit key 1, the output of comparator 406 goes active (high). In the first mode, control signal 16BITGR is high (and the output of NOR gate 409 is consequently high) and an active output from comparator 406 is gated through AND gate 410. The output of AND gate 410 is passed to AND gate 411 and gated with the output from the pixel comparison circuitry 305. The output of AND gate 411 goes directly to the "B" control input of selector 231 (in this embodiment multiplexer 304 and register 306 are eliminated). Thus, when the 8-bit graphics pixels output from attribute controller 233 of graphics backend 205 matches the 8-bit color key 1 and the window has been reached as determined by pixel comparison circuitry 305, the pixel data output from video backend 204 are passed through selector 231.

Certificate of Correction Change: Column 12, line 33, after "window" insert ",".

Column 11, beginning line 61 through Column 12, line 35 has been reproduced as follows:

FIG. 5 is an expanded functional block diagram of the video window position control circuits 308 and a corresponding portion of the gating of pixel position compare circuitry 305. Each position control circuit 310/312 is coupled to a screen position x-register 500 and a screen position y-register 501, and includes a screen x-position counter 502, and a screen y-position counter 503. In the preferred embodiment, registers 500 and

501 are located within video window control registers 203. For the window corresponding to the given video window control circuitry 308 or 310, registers 500 and 501 are loaded with a value representing the x and y screen position of the pixel in the upper left corner of that window (the starting pixel). Screen x-register 500 and screen y-register 501 in the preferred embodiment are loaded by CPU 101. The screen x-position counter 502 counts down from the value held in screen x-register 500 with each video clock when P is high for each display line and resets with each display horizontal synchronization signal (HSYNC) (Note that when P is high the CRT count matches the position count). Screen y-position counter 503 counts down from the value set into screen y-register 501 for each horizontal sync signal (HSYNC) at the start of each display line and resets with each VSYNC at the start of each new screen (The position counters are allowed to count only when they match their perspective CRT). The counts values in the counters of CRT position control circuitry 309 are compared pixel by pixel with the counts in screen x-position counter 502 and screen y-position 503 of each video window position control circuitry 308 and 310. When both the x and y counts in the counters of CRT position control circuitry 309 match the corresponding x and y counts in respective counters 502 and 503 of either video window control circuitry 308 or 310, the control signal P to multiplexer 304 is activated. The activation of control signal P indicates that the raster scan on display 106 has reached the position of a pixel within the window, and data from video pipeline 205 may be painted depending on the value being held in overlay OP Code (OOC) register 306 and the K control inputs to multiplexer 304.

Certificate of Correction Change: Column 12, line 67, delete "assumed:" and insert therefor --assumed):--

Column 12, lines 36-67 have been reproduced as follows:

A 4-bit OP Code loaded by CPU 101 into overlay OP Code register 306 in conjunction with the control signals applied to the "P" and "K" control inputs to multiplexer 304 control the presentation of an active (assumed high in the illustrated embodiment) control signal to the "B" control input to output multiplexer 231. The other ("A") input to output multiplexer 231 receives a bit from overlay mode register 402 (FIG. 4), as loaded by CPU 101. In the illustrated embodiment, the selection between the

streams from the graphics and video backends at the 0,1,2 inputs to output multiplexer 304 in response to the signals presented at the corresponding control inputs "A" and "B" is in accordance with Table II:

Control Input A	Control Input B	Selected Stream
0	0	Graphics or video pixels from graphics pipeline 205
1	0	Graphics pixels from CLUT 234 at input 1
0	1	Video or
1	1	graphics from video backend 204

The OP Codes used in the illustrated embodiment, the effective overlay and the corresponding inputs to the control inputs of multiplexer 304 are listed in Table III (active state is assumed):

Certificate of Correction Change: Column 13, line 55, after "for" delete "a".

Column 13, lines 48-67 have been reproduced as follows:

Display control circuits embodying the principles of the present invention have substantial advantages over the prior art. In particular, output control circuits built in accordance with the principles of the present invention allow for the flexible display of both graphics and video on the same screen. In particular, pixel position comparison circuitry 305 along with video window position control circuits 308 and 310 and CRT position control circuitry 309 allow for one or more windows from off-screen memory to be generated in specific areas of a display screen to the exclusion of any simultaneously generated data from on-screen memory. Further, color comparison circuitry 302 operating in conjunction with an overlay color key written into overlay color key register 303 allows window data to be presented on the display screen, to the exclusion of any concurrently generated graphics data, without the need for precise x- and y-position data for the window. Finally, the use of the graphics data from the graphics pipeline 205 to